# 实验6：实现路由算法

## Dijkstra 算法的思想

Dijkstra算法采用的是一种贪心的策略，声明一个数组dis来保存源点到各个顶点的最短距离和一个保存已经找到了最短路径的顶点的集合：T，初始时，原点 s 的路径权重被赋为 0 （dis[s] = 0）。若对于顶点 s 存在能直接到达的边（s,m），则把dis[m]设为w（s, m）,同时把所有其他（s不能直接到达的）顶点的路径长度设为无穷大。初始时，集合T只有顶点s。

然后，从dis数组选择最小值，则该值就是源点s到该值对应的顶点的最短路径，并且把该点加入到T中，OK，此时完成一个顶点，

然后，我们需要看看新加入的顶点是否可以到达其他顶点并且看看通过该顶点到达其他点的路径长度是否比源点直接到达短，如果是，那么就替换这些顶点在dis中的值。

然后，又从dis中找出最小值，重复上述动作，直到T中包含了图的所有顶点。

## Floyd 算法的思想

通过Floyd计算图G=(V,E)中各个顶点的最短路径时，需要引入两个矩阵，矩阵S中的元素a[i][j]表示顶点i(第i个顶点)到顶点j(第j个顶点)的距离。矩阵P中的元素b[i][j]，表示顶点i到顶点j经过了b[i][j]记录的值所表示的顶点。

假设图G中顶点个数为N，则需要对矩阵D和矩阵P进行N次更新。初始时，矩阵D中顶点a[i][j]的距离为顶点i到顶点j的权值；如果i和j不相邻，则a[i][j]=∞，矩阵P的值为顶点b[i][j]的j的值。 接下来开始，对矩阵D进行N次更新。第1次更新时，如果”a[i][j]的距离” > “a[i][0]+a[0][j]”(a[i][0]+a[0][j]表示”i与j之间经过第1个顶点的距离”)，则更新a[i][j]为”a[i][0]+a[0][j]”,更新b[i][j]=b[i][0]。 同理，第k次更新时，如果”a[i][j]的距离” > “a[i][k-1]+a[k-1][j]”，则更新a[i][j]为”a[i][k-1]+a[k-1][j]”,b[i][j]=b[i][k-1]。更新N次之后，操作完成！

## Dijkstra 算法伪代码

1. //  初始化,设从0开始
2. for i=[0,n)
3. dist[i] = map[0][i]
4. visit[0] = true;
6. for i=[1,n)
7. //  寻找最短路径(s,t)，同时把t加入S集合
8. min = MAX\_VALUE
9. for j=[0,n)
10. if !visit[j] && dist[j]<min
11. min = dist[j]//记录最小值和最小值的下标
12. min\_j = j
14. visit[j] = true
16. //  松弛边(t,v)，其中v为顶点
17. for k=[0,n)
18. if !visit[k] && dist[k]>dist[j]+tab[j][k]
19. dist[k] = dist[j]+tab[j][k]

## Floyd 算法伪代码

1. 枚举顶点k ∈ [1,n]
2. 以顶点k为中介点，枚举所有顶点对i和j（i ∈ [1,n],j ∈1[1,n]）
3. 如果dis[i][k] + dis[k][j] <dis[i][j]成立
4. 赋值dis[i][j] = dis[i][k] + dis[k][j]

## Dijkstra 算法性能分析

时间复杂度： O(n^2)

空间复杂度： O(n)

可优化的地方： 该算法在选择当前最小cost的节点时，可以考虑用堆排序进行优化。

## Floyd 算法性能分析

时间复杂度： O(n^3)

空间复杂度： O(n^2)

可优化的地方： 优化空间